Report

on

Face Recognition based E-Attendance System

Guided by:

Dr. Sanjeev Sharma

Dr. Anagha Khiste

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, PUNE

Title

Face Recognition Based E-Attendance System : A YOLO-v3 implementation

Introduction

A big amount of time of the classroom lecture gets dedicated to attendance maintenance which could, on the other hand be used for teaching purposes. Proxy attendance and miss attendance is another big issue which needs to be tackled. The biometric attendance system can't be implemented for every classroom and maintained without malfunctioning, and the process in addition would be time consuming.

Hence we came up with this idea: Building an E-attendance system which can detect the presence of students in a classroom.

This facilitates:

- Prevention of proxy and missing attendance.
- Ease in record maintenance.
- One click step to access the records.

Problem Statement

Development of a deep learning based model to recognize faces. Linking it with a database such as mongoDB or MySQL. Deploying it as a web-app or mobile app to enhance its accessibility. Testing its effectiveness in the classroom, to test its viability.

Objectives

- Development of a Yolo based Model for facial detection.
- Linking it with MySQL database for data maintenance and access.
- Deploying it as a Web-App or Mobile App with the help of containers.

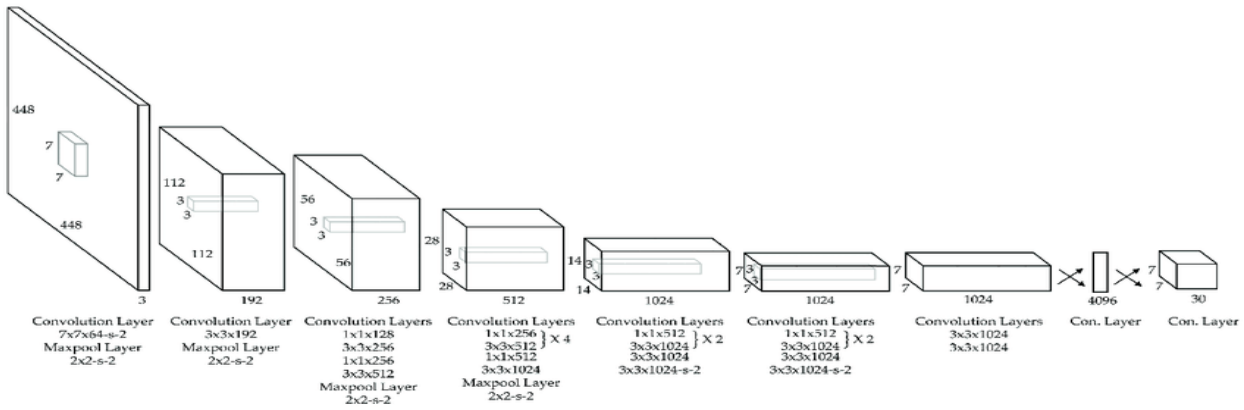
- Use of 4-5 students initially in the dataset for testing.
- Using a mobile device to monitor the attendance finally with just a click.

Methodology

The project is based on the YOLOv3 Darknet Model (AlexeyAB) which is the forked version of the original darknet model by Joseph Redmon. YOLO means You Only Look Once. It has state of the art abilities and proves to be one of the most efficient models when it comes to real time object detection.

Here, basically we have an image as an input, which goes through a ConvNet that results in a vector of features fed to a softmax to classify the object. Now, if we want to localize those objects in the image as well, we change the neural network to have a few more output units that encompass a bounding box. But Classification with Localization problems is unable to detect all the objects in the image since classification yields a single output class. To overcome this drawback , we can use a more robust technique which is ‘Object Detection’, using the YOLOv3 darknet model.

YOLO is a convolutional neural network (CNN). It is a stack made up of several units of convolution layers, leaky ReLU layers, max pool layers and fully connected layers. It uses the mean squared error loss function.



We start with placing a grid on top of the input image. (Commonly, number of grids is19x19). Every grid now determines if any object(based on the classes used) is present or absent(pc =0 or 1) where pc is the probability that a class/object exists. Next, it identifies the class to which the object in the grid belongs. It outputs the confidence scores for each and every class (c1 ,c2 ,c3 .....). The algorithm creates 'm' number of bounding boxes(Usually, m=5) for every grid in the image. Each bounding box has the coordinates of the top-left corner (bx ,by) along with its height and width (bh ,bw). Hence, the output 'Y' can be visualized as follows:

$$Y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

It outputs precise positions of bounding boxes, as the values bx, by, bh, bw are

computed relative to the cell. So, the finer grid we have the more precision we can obtain and also we have fewer chances of having multiple objects within a cell. The bounding box prediction of whether an object is present or not is done using Logistic Regression. For prediction of class, independent logistic classifiers are used instead of softmax layer. During training we use binary cross-entropy loss for the class predictions.

Intersection over Union (IoU)

This will measure if the object detection algorithm is working well. It computes the intersection over the union of the detected bounding box and the correct one. We identify a benchmark and consider an accurate object detection if the result of IoU is above that specific value (i.e. IoU <= 0.5). Here, if we have higher IoU value, then more accurate results we have.

$$IoU = \frac{size\ of\ the\ intersection\ area}{size\ of\ the\ union\ area}$$

Non-max suppression

YOLO could detect an object multiple times. To avoid that, we follow the following steps: First, we assign a probability on each detection, then we take the "largest probability" box. We now look at the boxes that overlap the most with the "largest probability" box and remove the ones that have high IoU (so the ones that have a big area of intersection). Finally, the remaining box is the correct detection. For multiple classes (objects), we implement non-max suppression independently

for each one.

Anchor boxes

For multiple objects in the same cell, Anchor boxes help us to overcome this issue. The idea here is to predefine different shapes (called anchor boxes) for each object and associate predictions to each one of them. Our output label now will contain 8 dimensions for each of the anchor boxes we predefined. Now, each object in the training image is assigned to the grid cell that contains that object's midpoint and the anchor box for the grid cell with highest IoU. The only thing it can not handle well is in case two objects in the same cell have the same anchor box. Additionally, we get to choose and redefine the shape of the anchor boxes.

$$Y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p'_c \\ b'_x \\ b'_y \\ b'_h \\ b'_w \\ c'_1 \\ c'_2 \\ c'_3 \end{bmatrix}$$

Feature Extractor

Darknet 53 feature extractor has 53 convolutional layers and is used in YOLOv3. Each of these layers is followed by batch normalization layer and leaky ReLU layer.We made use of the pre-trained weights darknet53.conv.74 for transfer learning, which is a powerful feature extractor. This network structure better

utilizes the GPU, making it more efficient to evaluate and thus faster.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Training

Restating, the approach that we have used for training is transfer learning. I have used the pre-trained weights of darknet53.conv.74 for fine-tuning the model.

During training the weights were saved in the backup folder after every 100 iterations. Also, after every 1000 iterations, the weights were stored, after which the detection of the objects using the latest weights were obtained in the backup folder.

Preliminary Literature Review

Presence of pre-existing works:

- Fingerprint Based Systems
- RFID Tags based Systems
- Semi Automated Systems
- CCTV based Attendance systems

From online resources:

- Mask R-CNN(Instance Segmentation, Object Localization and Object Detection).
- Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks: Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun
- AlexeyAB's guide to Yolo and Darnet
- PjReddie's guide to Yolo - Real Time Object Detection on github.com

Research Gaps

- YOLO imposes strong spatial constraints. Each grid cell is responsible for detection of only one object. Hence, it cannot detect multiple objects in the same grid cell. It struggles with small objects that appear in groups, such as swarms of bees.
- Available research mostly treats the object as a particle, thereby leaving out the more complicated dynamics.
- Faces would have to get recognised in all sorts of varying conditions and complex environments.
- Although the Yolo model detects the faces quite accurately, using focal loss, it drops the mAP by 2, hence the predictions for separate objectness and conditional class may vary.

Work Done

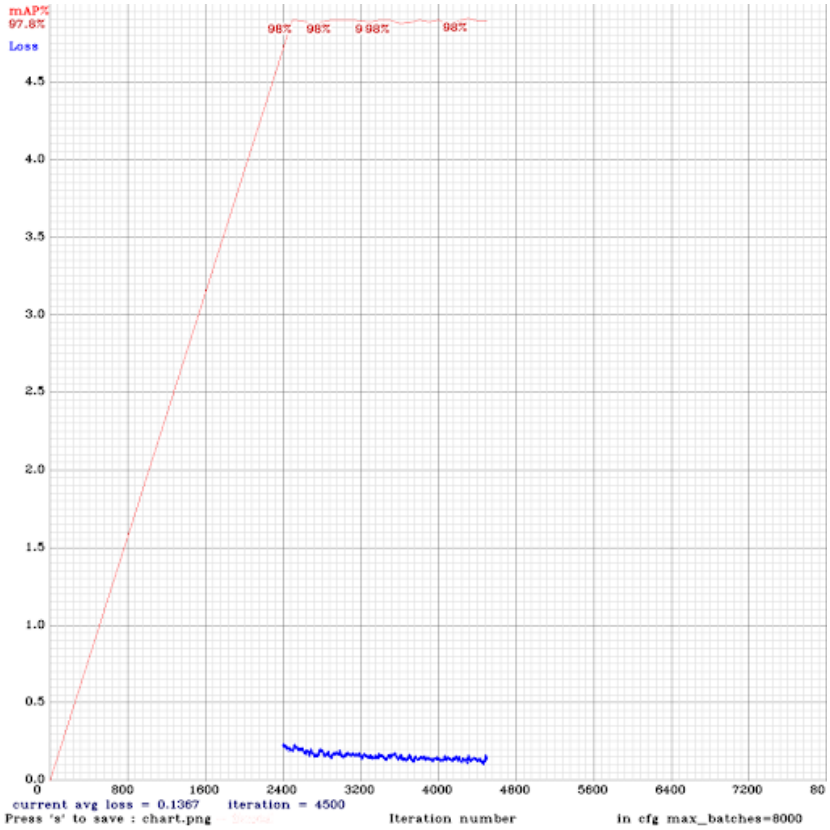
The results shown below are evaluated just before 4500 iterations.

Test Image:

- Output Predictions for each class present in the test image:

```
./build/darknet/x64/data/obj/imgtest52.jpg: Predicted in 1238.291000 milli-seconds.
Abhi: 95%
Karan: 99%
```

- We could also draw mAP-chart (red-line) in the Loss-chart Window. mAP will be calculated for each 4 Epochs, where 1 Epoch = images_in_train_txt / batch iterations. The Loss-chart Window is shown below for 4500 iterations:



Loss-chart Window for mAP

- Output Predictions for each class present in the test image, including the coordinates of each bounding box:

```
./build/darknet/x64/data/obj/imgtest52.jpg: Predicted in 266.789000 milli-seconds.
Abhi: 95%       (left_x:  281   top_y:  178   width:   26   height:   34)
Karan: 99%      (left_x:  407   top_y:  208   width:   36   height:   39)
```

- Checking Accuracy mAP at the IoU value 50, we get:

```
 calculation mAP (mean average precision)...
56
 detections_count = 118, unique_truth_count = 98
class_id = 0, name = Shubham, ap = 95.33%        (TP = 30, FP = 4)
class_id = 1, name = Karan, ap = 100.00%         (TP = 22, FP = 0)
class_id = 2, name = Abhi, ap = 95.84%           (TP = 22, FP = 0)
class_id = 3, name = Saurabh, ap = 100.00%       (TP = 21, FP = 0)

 for conf_thresh = 0.25, precision = 0.96, recall = 0.97, F1-score = 0.96
 for conf_thresh = 0.25, TP = 95, FP = 4, FN = 3, average IoU = 81.29 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.977926, or 97.79 %
Total Detection Time: 32 Seconds
```

- Generated a text file which stores all the results of the detection phase using all the test images. The below screenshot has been taken from the file - “result.txt” :

```
Enter Image Path: build/darknet/x64/data/obj/img62.jpg: Predicted in 19.196000 milli-seconds.
Enter Image Path: build/darknet/x64/data/obj/img282.jpg: Predicted in 19.114000 milli-seconds.
Shubham: 100%   (left_x:  204   top_y:  319   width:   56   height:   71)
Saurabh: 66%    (left_x:  468   top_y:  261   width:   26   height:   28)
Enter Image Path: build/darknet/x64/data/obj/img99.jpg: Predicted in 19.212000 milli-seconds.
Saurabh: 100%   (left_x:  354   top_y:  162   width:   53   height:   68)
Enter Image Path: build/darknet/x64/data/obj/img42.jpg: Predicted in 19.213000 milli-seconds.
Shubham: 98%    (left_x:  306   top_y:  293   width:   31   height:   42)
Saurabh: 97%    (left_x:  356   top_y:  272   width:   23   height:   35)
Abhi: 74%       (left_x:  514   top_y:  219   width:   28   height:   44)
Enter Image Path: build/darknet/x64/data/obj/img411.jpg: Predicted in 19.343000 milli-seconds.
Saurabh: 97%    (left_x:  105   top_y:  221   width:   34   height:   49)
Shubham: 100%   (left_x:  292   top_y:  226   width:   39   height:   42)
Abhi: 97%       (left_x:  386   top_y:  228   width:   39   height:   46)
Karan: 100%     (left_x:  596   top_y:  298   width:   58   height:   69)
Enter Image Path: build/darknet/x64/data/obj/img580.jpg: Predicted in 19.290000 milli-seconds.
Shubham: 100%   (left_x:  178   top_y:  194   width:  231   height:  205)
Enter Image Path: build/darknet/x64/data/obj/img520.jpg: Predicted in 19.121000 milli-seconds.
Karan: 98%      (left_x:  189   top_y:  251   width:   41   height:   50)
Saurabh: 99%    (left_x:  607   top_y:  173   width:   30   height:   48)
Enter Image Path: build/darknet/x64/data/obj/img287.jpg: Predicted in 19.168000 milli-seconds.
Saurabh: 100%   (left_x:  358   top_y:  264   width:   58   height:   77)
Enter Image Path: build/darknet/x64/data/obj/img258.jpg: Predicted in 19.308000 milli-seconds.
Abhi: 100%      (left_x:  376   top_y:  244   width:   30   height:   47)
```

- After increasing the no. of iterations to 5000, we got better results which could is shown as:

```
Enter Image Path: build/darknet/x64/data/obj/img639.jpg: Predicted in 20.302000 milli-seconds.
Saurabh: 100%   (left_x:  329   top_y:  246   width:   52   height:   72)
Enter Image Path: build/darknet/x64/data/obj/img485.jpg: Predicted in 20.375000 milli-seconds.
Karan: 96%      (left_x:  206   top_y:  231   width:   38   height:   44)
Abhi: 100%      (left_x:  298   top_y:  222   width:   35   height:   42)
Shubham: 100%   (left_x:  464   top_y:  200   width:   40   height:   50)
Enter Image Path: build/darknet/x64/data/obj/img595.jpg: Predicted in 20.233000 milli-seconds.
Shubham: 99%    (left_x:  359   top_y:  250   width:   88   height:  112)
Saurabh: 100%   (left_x:  523   top_y:  161   width:   27   height:   38)
Enter Image Path: build/darknet/x64/data/obj/img407.jpg: Predicted in 20.235000 milli-seconds.
Saurabh: 87%    (left_x:  425   top_y:  224   width:   23   height:   29)
Enter Image Path: build/darknet/x64/data/obj/img605.jpg: Predicted in 20.394000 milli-seconds.
Shubham: 100%   (left_x:   71   top_y:  259   width:   68   height:   89)
Karan: 93%      (left_x:  384   top_y:  214   width:   25   height:   30)
Abhi: 99%       (left_x:  705   top_y:  240   width:   48   height:   63)
Enter Image Path: build/darknet/x64/data/obj/img332.jpg: Predicted in 20.390000 milli-seconds.
Saurabh: 100%   (left_x:  410   top_y:  236   width:   36   height:   50)
Enter Image Path: build/darknet/x64/data/obj/img575.jpg: Predicted in 20.503000 milli-seconds.
Saurabh: 100%   (left_x:  389   top_y:  114   width:   60   height:   64)
Enter Image Path: build/darknet/x64/data/obj/img374.jpg: Predicted in 20.632000 milli-seconds.
Saurabh: 99%    (left_x:  309   top_y:  250   width:   30   height:   37)
```

- After parsing the file to the required format, we made a text file containing names of students present on a day, suppose for date 06-02-2020(a part of that file has been attached below), then we first used all required operations to extract the names of the students from that file and then made a database using SQL, where the extract of the DB Browser has also been shown below. Also, the file containing the code to parse the file and for creating the database has also been shown.

```
Path: Enter Image build/darknet/x64/data/obj/img23.jpg:
(left_x: Shubham: 100% 366
Path: Enter Image build/darknet/x64/data/obj/img405.jpg:
(left_x: Abhi: 100% 295
Path: Enter Image build/darknet/x64/data/obj/img621.jpg:
(left_x: Saurabh: 100% 217
Path: Enter Image build/darknet/x64/data/obj/img292.jpg:
(left_x: Abhi: 100% 268
Path: Enter Image build/darknet/x64/data/obj/img511.jpg:
(left_x: Shubham: 99% 342
(left_x: Abhi: 95% 379
(left_x: Saurabh: 95% 606
Path: Enter Image build/darknet/x64/data/obj/img237.jpg:
(left_x: Karan: 100% 123
Path: Enter Image build/darknet/x64/data/obj/img77.jpg:
(left_x: Karan: 91% 369
Path: Enter Image build/darknet/x64/data/obj/img506.jpg:
(left_x: Saurabh: 84% 406
Path: Enter Image build/darknet/x64/data/obj/img437.jpg:
(left_x: Saurabh: 96% 617
Path: Enter Image build/darknet/x64/data/obj/img302.jpg:
(left_x: Abhi: 97% 220
(left_x: Saurabh: 99% 286
(left_x: Shubham: 98% 680
```

SQLite Code to extract object names:

```
import sqlite3
from sqlite3 import Error
with open ('./BTP2020/Day1') as fo:
        set1=set()
        for rec in fo:
                set1.add(rec.split(' ')[1])
                if "Enter" in set1:
                        set1.remove("Enter")
                #print(set1)
# def sql_connection():
    # try:
        # con = sqlite3.connect('mydatabase.db')
        # return con
    # except Error:
        # print(Error)
con = sqlite3.connect('mydatabase.db')
def sql_table(con):
    cursorObj = con.cursor()
    cursorObj.execute("CREATE TABLE employees(id integer PRIMARY KEY, name text, date text)")
    con.commit()
def sql_insert(con, entities):
    cursorObj = con.cursor()
    cursorObj.execute('INSERT INTO employees(id, name, date) VALUES(?, ?, ?)', entities)
    con.commit()
j=1
for i in set1:
        entities = (j, i, '06-02-2020')
        sql_insert(con, entities)
        j=j+1
```

Sample Output:

	id	name	date
	Filter	Filter	Filter
1	1	Shubham:	06-02-2020
2	2	Saurabh:	06-02-2020
3	3	Karan:	06-02-2020
4	4	Abhi:	06-02-2020

References

1. Face Recognition based Attendance System
2. *AlexeyAB's guide to Yolo and Darnet*
3. Deep Neural Network for Human Face Recognition - MECS 2018
4. *PjReddie's guide to Yolo - Real Time Object Detection*
5. Image-based Face Detection and Recognition - F. Ahmad
6. Evaluation of Feature Extraction Techniques - IJSRST 2018.